

WRDC-TR-90-8007  
Volume VIII  
Part 37

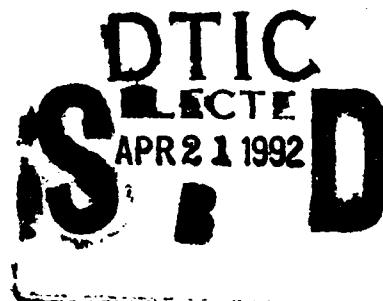
**AD-A248 976**



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)  
Volume VIII - User Interface Subsystem  
Part 37 - Layout Optimization System Development Specification

S. Barker, F. Glandorf

Control Data Corporation  
Integration Technology Services  
2970 Presidential Drive  
Fairborn, OH 45324-6209



September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

**92-10046**



MANUFACTURING TECHNOLOGY DIRECTORATE  
WRIGHT RESEARCH AND DEVELOPMENT CENTER  
AIR FORCE SYSTEMS COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

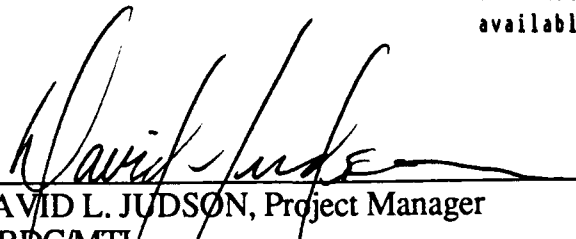
92 4 20 082

## NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.


This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

  
DAVID L. JUDSON, Project Manager  
WRDC/MTI  
Wright-Patterson AFB, OH 45433-6533

25 July 91  
DATE

FOR THE COMMANDER:

  
BRUCE A. RASMUSSEN, Chief  
WRDC/MTI  
Wright-Patterson AFB, OH 45433-6533

25 July 91  
DATE

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE					
1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None			
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) DS 620344800		5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VIII, Part 37			
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services		6b. OFFICE SYMBOL (if applicable)		7a. NAME OF MONITORING ORGANIZATION WRDC/MTI	
6c. ADDRESS (City, State, and ZIP Code) 2970 Presidential Drive Fairborn, OH 45324-6209		7b. ADDRESS (City, State, and ZIP Code) WPAFB, OH 45433-6533			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF		8b. OFFICE SYMBOL (if applicable) WRDC/MTI		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464	
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, Ohio 45433-6533		10. SOURCE OF FUNDING NOS.			
11. TITLE (In See block 19		PROGRAM ELEMENT NO. 78011F	PROJECT NO. 595600	TASK NO. F95600 WORK UNIT NO. 20950607	
12. PERSONAL AUTHOR(S) Structural Dynamics Research Corporation: Barker, S., Glandorf, F.					
13a. TYPE OF REPORT Final Report	13b. TIME COVERED 4 / 1 / 87 - 12 / 31 / 90	14. DATE OF REPORT (Yr., Mo., Day) 1990 September 30		15. PAGE COUNT 56	
16. SUPPLEMENTARY NOTES WRDC/MTI Project Priority 6203					
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify block no.)			
FIELD	GROUP				SUB GR.
1308	0905				
19. ABSTRACT (Continue on reverse if necessary and identify block number)  This specification establishes the performance, development, test and qualification requirements for the system identified as the Layout Optimization System.  BLOCK 11:  INTEGRATED INFORMATION SUPPORT SYSTEM Vol VIII - User Interface Subsystem  Part 37 - Layout Optimization System Development Specification					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified			
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson		22b. TELEPHONE NO. (Include Area Code) (513) 255-7371		22c. OFFICE SYMBOL WRDC/MTI	

### FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

<u>SUBCONTRACTOR</u>	<u>ROLE</u>
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS

	<u>Page</u>
SECTION 1.0 SCOPE .....	1-1
1.1 Identification .....	1-1
1.2 Functional Summary .....	1-1
SECTION 2.0 DOCUMENTS .....	2-1
2.1 Reference Documents .....	2-1
2.2 Terms and Abbreviations .....	2-1
SECTION 3.0 Requirements .....	3-1
3.1 Computer Program Definition .....	3-1
3.1.1 System Capacities .....	3-1
3.1.2 Interface Requirements .....	3-1
3.1.2.1 Interface Block Diagram .....	3-1
3.1.2.2 Detailed Interface Definition .....	3-2
3.1.2.2.1 Callable Routine Interface .....	3-2
3.1.2.2.2 Form Processor Interface .....	3-3
3.1.2.2.3 Chart Knowledge Base Interface ...	3-3
3.2 Detailed Functional Requirements .....	3-4
3.2.1 Chart, Object and Relation Archetypes	3-4
3.2.2 Archetype Definition .....	3-5
3.2.2.1 Building Charts .....	3-6
3.2.2.1.1 Defining Objects and Relations .....	3-6
3.2.2.1.2 Deleting Objects and Relations .....	3-7
3.2.3 Object and Relationship Forms .....	3-7
3.2.4 Chart Layout .....	3-9
3.2.4.1 Layout Objectives .....	3-9
3.2.4.2 Layout Algorithm .....	3-11
3.2.5 Chart Display .....	3-18
3.2.6 Chart Knowledge Base .....	3-19
3.2.6.1 Knowledge Base Utility Program .....	3-19
3.2.6.1.1 Chart Description .....	3-19
3.2.6.1.2 Object Description .....	3-20
3.2.6.1.3 Relation Description .....	3-22
3.2.6.2 Knowledge Base Description .....	3-23
3.2.6.2.1 Charts File Layout .....	3-23
3.2.6.2.2 Objects File Layout .....	3-24
3.2.6.2.3 Relations File Layout .....	3-24
3.3 Performance Requirements .....	3-24
3.3.1 Programming Methods .....	3-24
3.3.2 Program Organization .....	3-25
3.3.3 Expandability .....	3-25

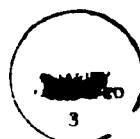
TABLE OF CONTENTS, continued

3.3.4	Error Recovery .....	3-25
3.4	Data Base Requirements .....	3-25
3.5	Adaptation Requirements .....	3-25
SECTION 4.0	QUALITY ASSURANCE PROVISIONS .....	4-1
4.1	Introduction and Definition .....	4-1
4.2	Compute Programming Test and Evaluation	4-1
SECTION 5.0	PREPARATION FOR DELIVERY .....	5-1

APPENDICES

APPENDIX A	CALLABLE ROUTINE DESCRIPTIONS .....	A-1
APPENDIX B	STRETCHY LINES .....	B-1
APPENDIX C	STRETCHY LINE EXAMPLE .....	C-1

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
3-1	Interface Block Diagram .....	3-2
3-2	Veritcal and Horizontal Orientations .....	3-4
3-3	Symbols .....	3-5
3-4	Diagrams A, B, C .....	3-10
3-5	Object 5 Added to Object 1 .....	3-11
3-6	Object 5 related to Object 2 .....	3-12
3-7	Related Objects .....	3-13
3-8	Layout Adjustments .....	3-14
3-9	Entire Object Adjustments .....	3-13
3-10	Object Diagonal .....	3-14
3-11	Independent group of Objects .....	3-15
3-12	Object X .....	3-16
3-13	Permitted Adjustments .....	3-17
3-14	Line Length Rules .....	3-18
3-15	Chart Description .....	3-20
3-16	Object Description .....	3-21
3-17	Relation Description .....	3-22

## SECTION 1

### SCOPE

#### 1.1 Identification

This specification establishes the performance, development, test and qualification requirements for the system identified as the Layout Optimization System or alternatively as the Layout System.

#### 1.2 Functional Summary

This Computer Program Component (CPC) consists of several parts. A portion will be within the Form Processor Configuration Item; a portion will be part of the Application Interface Configuration Item; and, another portion will comprise a data base of information to govern the execution.

The major functions of the Layout Optimization System capability are:

- o The compile-time definition of the structure of objects which will be manipulated by the Layout Optimization System.
- o The compile-time specification of the pictorial representation of objects which will be manipulated by the Layout system.
- o The run-time creation or modification of such objects.
- o The automatic positioning of such objects based upon built-in criteria.
- o The run-time deletion of objects.
- o The run-time definition or modification of relationships between objects.
- o The run-time specification of the data associated with the structure of the objects being manipulated.



## SECTION 2

### DOCUMENTS

#### 2.1 Reference Documents

- [1] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620244200, 24 October 1986.
- [2] Systran, ICAM Documentation Standards, IDS 150120000C, 15 September 1983.
- [3] Structural Dynamics Research Corporation, Optimization of Forms Layout Requirements Document, 24 February 1987.
- [4] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS 620244401, 24 October 1986.
- [5] Structural Dynamics Research Corporation, C Coding Guidelines, 16 July 1984.
- [6] Structural Dynamics Research Corporation, Rapid Application Generator Development Specification, DS 620244502A, 24 October 1986.

#### 2.2 Terms and Abbreviations

Application Generator: (AG), subset of the IISS User Interface that consists of software modules that generate IISS application code and associated form definitions based on a language input. The part of the AG that generates report programs is called the Report Writer. The part of the AG that generates interactive applications is called the Rapid Application Generator.

Application Interface: (AI), subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process: (AP), a cohesive unit of software that can be initiated as a unit to perform some function or functions.

Archetype: the original pattern or model of which all things of the same type are copied.

Attribute: field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Common Data Model: (CDM), IISS subsystem that describes common data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Computer Program Configuration Item: (CPCI), an aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.

Conceptual Schema: (CS), the standard definition used for all data in the CDM. It is based on IDEF1 information modeling.

Device Drivers: (DD), software modules written to handle I/O for a specific kind of terminal. The modules map terminal-specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: is similar to the open list, except that it contains only those forms that have been added to the screen and are currently displayed on the screen.

External Schema: (ES), an application's view of the CDM's conceptual schema.

Field: two dimensional space on a terminal screen.

Form: structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows.

Form Definition: (FD), forms definition language after compilation. It is read at run-time by the Form Processor.

Forms Definition Language: (FDL), the language with which electronic forms are defined.

Forms Driven Form Editor: (FD FE), subset of the FE which consists of a forms driven application used to create Form Definition files interactively.

Form Editor: (FE), subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor and the Forms Language Compiler.

Form Hierarchy: a graphic representation of the way in which forms, items and windows are related to their parent form.

Forms Language Compiler: (FLAN), subset of the FE that consists of a batch process that accepts a series of forms definition language statements and produces form definition files as output.

Form Processor: (FP), subset of the IISS User Interface that consists of a set of callable execution time routines available to an application program for form processing.

Icon: a pictorial representation.

IISS Function Screen: the first screen that is displayed after logon. It allows the user to specify the function he wants to access and the device type and device name on which he is working.

Integrated Information Support System: (IISS), a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item: non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output.

Message: descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: a line on the terminal screen that is used to display messages.

Network Transaction Manager: (NTM), IISS subsystem that performs the coordination, communication and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Neutral Data Manipulation Language: (NDML), the command language by which the CDM is accessed for the purpose of extracting, deleting, adding, or modifying data.

Operating System: (OS), software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: instance of forms in windows that are created whenever a form is added to a window.

Paging and Scrolling: a method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Physical Device: a hardware terminal.

Presentation Schema: (PS), may be equivalent to a form. It is the view presented to the user of the application.

Qualified Name: the name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

Report Definition Language: (RDL), an extension of the Forms Definition Language that includes retrieval and calculation of database information and is used to define reports.

Report Writer: (RW), part of the Application Generator that generates source code for report programs based on a language input.

Subform: a form that is used within another form.

Text Editor: (TE), subset of the IISS User Interface that consists of a file editor that is based on the text editing functions built into the Form Processor.

User Data: data which is either input by the user or output by the application programs to items.

User Interface: (UI), IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System: (UIDS), collection of IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor and the Application Generator.

User Interface Management System: (UIMS), the runtime UI. It consists of the Form Processor, Virtual Terminal, Application Interface, the User Interface Services and the Text Editor.

User Interface Services: (UIS), subset of the IISS User Interface that consists of a package of routines that aid users in controlling their environment. It includes message management, change password, and application definition services.

User Interface/Virtual Terminal Interface: (UI/VTI), another name for the User Interface.

Window: dynamic area of a terminal screen on which predefined forms may be placed at run time.

Window Manager: a facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor.

SECTION 3  
REQUIREMENTS

3.1 Computer Program Definition

The Layout Optimization System User Interface supports application programs which manipulate objects of a more general nature than current form fields. This support includes the automatic positioning of these objects according to built-in criteria. The most common applications are IDEF or structure charts. This functionality is an extension of the current User Interface.

3.1.1 System Capacities

The Layout System will operate in the IISS environment on those hosts supported by IISS.

It will be available on those output devices which are supported by the User Interface within the IISS environment.

3.1.2 Interface Requirements

The Layout System will be compatible with the host operating systems - VMS on the VAX and MVS on the IBM.

The Layout System provides an interface to application programs through a library of callable routines.

3.1.2.1 Interface Block Diagram

The interface block diagram for the Layout Optimization System is illustrated in Figure 3-1.

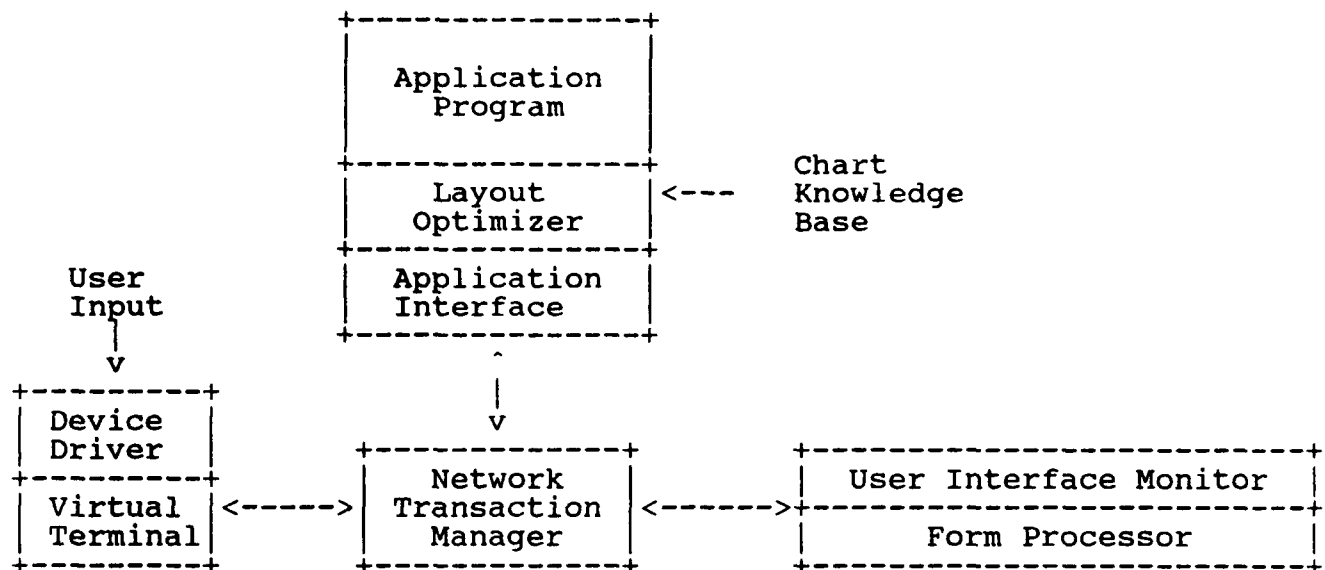


Figure 3-1 Interface Block Diagram

### 3.1.2.2 Detailed Interface Definition

The Layout Optimization System is embedded within the User Interface System. An application program accesses the facilities of this system through the Layout system's application interface. The components of the Layout system's application interface are: 1) an interface for the callable routines, and 2) an interface to the Form Processor. In addition to the application interface, the Layout system may make use of a chart knowledge base.

#### 3.1.2.2.1 Callable Routine Interface

An application program accesses the Layout system through a library of callable routines. These routines are detailed in Appendix A. These routines have both input and output parameters and will return to the application program a completion status code. The callable routines facilitate the definition of objects including their names and types. They facilitate the definition of relationships and their names and types and the names of the objects being related.

#### 3.1.2.2.2 Form Processor Interface

The objects of a chart are pictorially represented by forms. The object's pictorial form definition, a form definition object file created by FLAN, must be made prior to utilizing the Layout system. During the building of a chart, the forms associated with the objects being defined will be accessed by the Layout system through Form Processor calls, such as ADDFRM. The object forms will be placed on the display list for the chart being defined.

Each chart will have its own display list. This technique has been chosen to avoid possible naming conflicts when building charts. In order for the application program to provide textual data to the object forms, the application program must have access to the display list for the chart. The Layout system provides the interface for accessing the desired chart's display list. A callable routine is provided which will allow the application program to switch display lists from the application program's own display list to the display list of the chart. Another routine is provided to return the application program to its own display list.

#### 3.1.2.2.3 Chart Knowledge Base Interface

The Layout system also makes use of a chart knowledge base. The knowledge base for charts contains the information required by the Layout system to define the chart, object and relation archetypes comprising that chart. A chart knowledge base may be created and maintained by using the chart knowledge base utility program (see Section 3.2.6). The Layout system will interface to the information of the knowledge base. A knowledge base for charts is not required. Charts, objects and relations can be completely defined through the Layout system's callable routines.



### 3.2 Detailed Functional Requirements

The Layout Optimization System is an application callable system that uses features of the User Interface to draw charts. The system views charts as groups of objects and relationships between the objects. Each chart may have a knowledge base that describes the type of objects contained in the chart and names the form definition file containing the pictorial representation of the object. The knowledge base also contains the name of each relation type for the chart. The name associates a pictorial representation consisting of a stretchy line, originator and terminator symbols to the relation.

#### 3.2.1 Chart, Object and Relation Archetypes

Every chart has associated with it certain characteristics such as object orientation, as seen in Figure 3-2.

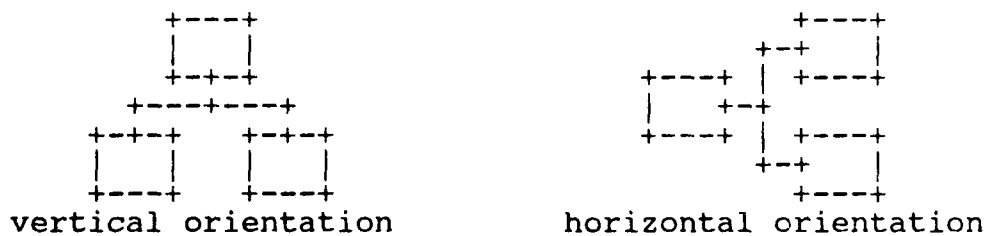


Figure 3-2. Vertical and Horizontal Orientations

Likewise, the objects and relations that comprise a chart have their respective characteristics. For example, an object, may require that all relationships for which it is the parent or owner must emanate diagrammatically from the bottom of the object. Similarly, a relation may require that it must terminate at the top of the object which is the child of the relationship. These characteristics or archetypes provide the basis for the generation of a chart and the creation of the objects and relationships which make up the chart.

### 3.2.2 Archetype Definition

The Layout system is accessed through callable routines. Details of the Layout system function calls can be found in Appendix A.

The first routine called must be INITFL. This routine allocates memory and initializes the Layout data structures in preparation for further calls.

The DCHART routine is used to define the chart archetype at run time. The archetype description for a chart includes the desired object orientation, such as vertical or horizontal and the initial diagonal layout preference such as downward from the left or upward from the left. If a knowledge base is to be used, this routine is not required.

The routine which describes an object archetype is called DOBJTP. This routine provides for naming the object archetype and specifying the name of the form to be used for the pictorial representation of any objects patterned after this archetype.

A relation archetype is described through the DRELTP service call. This routine provides for naming the relation archetype and specifying the names of the forms used to pictorially represent the connector termination and origination symbols (Figure 3-3).

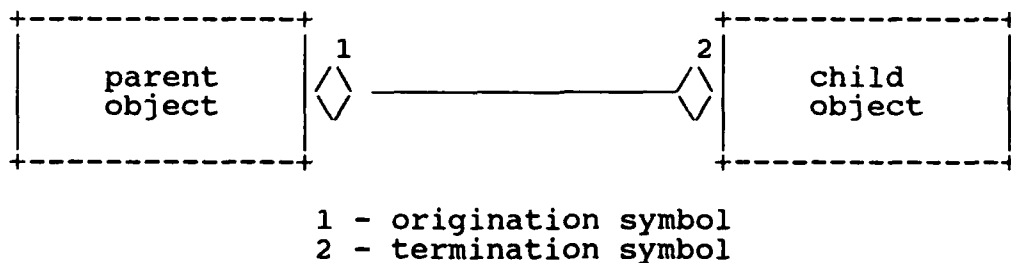


Figure 3-3

It also allows you to specify the line style to use for the connector and the form to use if the relation is complex, such as the IDEF1X generic relation. The existence of a complex connector type implies a one object to many object type of relation. A relation is assumed to be a one object to one object relation unless the complex connector is specified.

### 3.2.2.1 Building Charts

After all the archetype descriptions have been made, the creation and population of the chart can begin. The archetype descriptions will be used as templates when an instance of a chart, object or relation is created.

#### 3.2.2.1.1 Defining Objects And Relationships

The Layout system provides three routine calls to create instances of the archetypes defined. The first of these is the MAKCHT routine. This routine has two functions. First, MAKCHT will search the chart knowledge base for the name specified in the chart archetype name parameter. If the chart is defined in the knowledge base, the chart, object and relation archetype information within the knowledge base is loaded into memory.

If the chart is not found in the knowledge base, then all object and relation archetype descriptions must be made through the callable routines.

Secondly, MAKCHT creates an instance of the named chart archetype and gives the chart instance the name specified. All further references to this chart must use the instance name. A display list is created for this chart.

The MAKOBJ routine creates an instance of the named object archetype and gives the object instance the name specified. All further references to this object must use the instance name. The form named in the archetype description is copied, using dynamic field calls, and given the name of the object instance. It is added to the chart's display list.

The MAKREL routine creates a relationship between the two named objects from the named relation archetype and gives the relationship the instance name specified. All further references to this relationship must use the instance name. The two objects must exist. An error will be returned otherwise. The MAKREL routine, using dynamic field calls, will create a form with the name of the relationship. The originator, terminator and complex symbol forms will be added to the relationship form. Creating the relationship form and adding the other forms to its definition provides path names for accessing the forms by the application program.

#### 3.2.2.1.2 Deleting Object And Relationships

Under some circumstances, it may be necessary for the application program to delete object, relation and chart instances. The Layout system provides this capability. The DELOBJ routine will delete the named object instance from the specified chart. In addition, the object will be removed from the chart's display list. The object will not be altered in any way, however, if it is defined as part of an active non-complex relationship.

The DELREL routine will delete the named relationship and remove the relationship form definitions from the display list for the designated chart. The object instances involved in the relationship will not be altered in any way. In the case of a complex relationship, the relationship will not be deleted until all child objects of the relationship have been deleted through separate DELREL routine calls.

The DELCHT routine will remove the entire chart instance definition from the Layout system. This includes all object and relationship instance definitions. The display list for the chart will be deleted and all computer resources allocated to the chart will be returned to the system.

#### 3.2.3 Object And Relationship Forms

The forms to be used to pictorially represent the objects and relationships of the chart are not readily accessible to the application program (section 3.1.2.2.2). The Layout system provides the means, (BEGCHT) to enable the application program to gain accessibility to the display lists for the forms for the purpose of providing text and otherwise manipulating the forms to suit the needs of the chart being defined.

BEGCHT sets the display list assignment for the application program to the display list for the chart specified in the call. Upon completion of this routine the application has access to the forms for the chart. All Form Processor calls are available to the application program for manipulating the forms for the chart until ENDCHT is called by the application program. ENDCHT resets the display list assignment for the application program to the application program's original display list.

An example of the need to use this facility can be seen in Appendix C. If the IDEF1X entity defined in Appendix C requires an additional key attribute and regular attribute the BEGCHT routine would be called first to gain access to the chart's display list. The SETDIM dynamic field call is used to increase by one the size of the array definitions for both key\_field and attr\_field. Text is supplied to these fields by using the PDATA Form Processor call.

The MAKCHT, MAKOBJ and MAKREL routines require ten character instance names. These names are used to reference the instances of charts, objects and relationships. There is a reason for the ten character limit on instance names. The instance name, in the case of MAKOBJ, will also be used to name the form created to pictorially represent the object. The application program can then reference the form by the instance name. The path name to an item of the object form would be "object\_instance\_name.item".

The MAKREL routine may create and name up to three forms during the make relationship process. The instance name for the relationship must then, by necessity, provide for three unique names. This will be accomplished by appending to the end of the instance name the letter "O" for "O"riginating, "T" for "T"erminating and "C" for "C"omplex to denote the form desired when using form processor calls. Therefore, the instance name supplied to the MAKREL routine must be nine or less characters. If it is ten characters the last character will be truncated. An example of the path name to an item of the originating symbol form would be "relationship\_instance\_nameO.item".

The GINSNM routine, generate instance name, is provided to generate instance names if the application program does not have the capacity to do so. The name returned to the application program will be nine characters, left justified in the ten character parameter field.

#### 3.2.4 Chart Layout

The Layout system provides a routine, DRWCHT, to layout or draw the chart which has been defined. Drawing the chart entails: 1) organizing the objects based upon the characteristics of the chart definition, 2) aligning the objects and relationship connector symbols in accordance with the characteristics of the object and relation definitions, and 3) manipulating the object and relationship forms in order to conform to the characteristics of the chart. It should be noted that drawing does not include displaying the chart. Displaying the chart is covered in Section 3.2.5.

##### 3.2.4.1 Layout Objectives

The Layout system provides the application program with the ability to construct virtually all types of charts. Specifying the appropriate characteristics for object and relation archetypes can produce hierarchy charts as well as network charts. However, if the objects and relations defined for a chart provide no characteristics to use as rules for laying out the chart, the system must make some assumptions about how the chart is to be organized.

To handle this circumstance, the Layout system has been designed with the following objectives in mind:

- o Provide a generalized layout functionality
- o Minimize line crossings
- o Minimize line bends

There are several basic layout patterns that can be used as a guide for organizing a chart. Figure 3-4 shows three such possibilities.

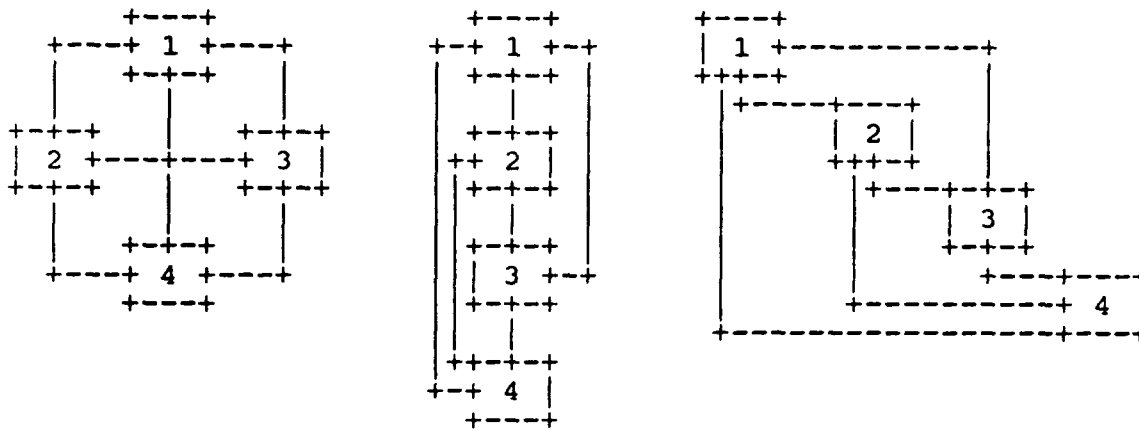


Diagram A

Diagram B

Diagram C

Figure 3-4. Diagrams A, B, C

Diagram A illustrates the general network approach to chart organization. It is easy to see that this technique will cause excessive line crossings. This will make a chart hard to read as the number of objects increase.

Diagram B, while demonstrating no line crossings with four objects, causes several lines to have two line bends. Additionally, if a fifth object is added to the diagram, line crossings will be created.

Diagram C shows no line crossings and each line has only one bend. It can be shown that by adding objects to this example, the number of line crossings will remain at a minimum. For this reason the Layout system has adopted the third approach to chart layout when the application program has provided no rules or guidelines in the archetype descriptions.

### 3.2.4.2 Layout Algorithm

The layout or drawing of the chart begins when DRWCHT is called. When the call is received, the Layout system locates the object possessing the most relationships for which it is the parent. Each object related to this parent object is found by examining the relationships for the parent object. The related objects are placed at the diagonals of the parent object. Objects are always placed at the least populated corner of the parent object. Figure 3-5 illustrates how object 5 will be added at the upper left corner of object 1.



Figure 3-5. Object 5 added to Object 1

If it is found that object 5 is related to object 2 as well as object 1, then object 5 will be placed at the upper right-hand corner of object 2 (see Figure 3-6). This is in keeping with the diagonal object orientation philosophy which states that all related objects should be placed along the same diagonal.



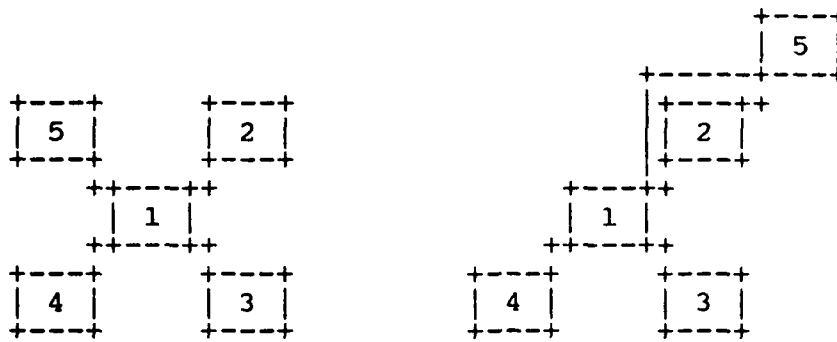


Figure 3-6. Object 5 Related to Object 2

As objects and relationships are added to the layout drawing it may come about that objects which were unrelated are now related. Figure 3-7 illustrates this possibility.

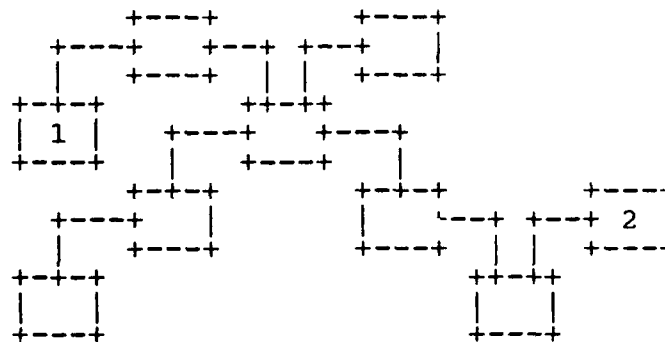


Figure 3-7. Related Objects

If object 1 and object 2 have a relationship established between them, some adjustments to the layout must be made if line crossings and line bends are to be minimized. The Layout system will make the adjustments illustrated in Figure 3-8. Objects 1 and 2 are adjusted so that they fall along the same diagonal.

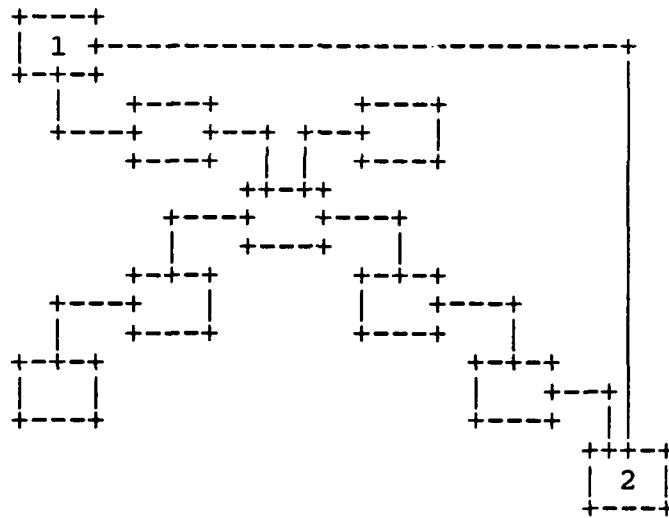


Figure 3-8. Layout Adjustments

More radical adjustments may have to be made, however. Figure 3-9 is an example of an entire group of objects which must be moved.

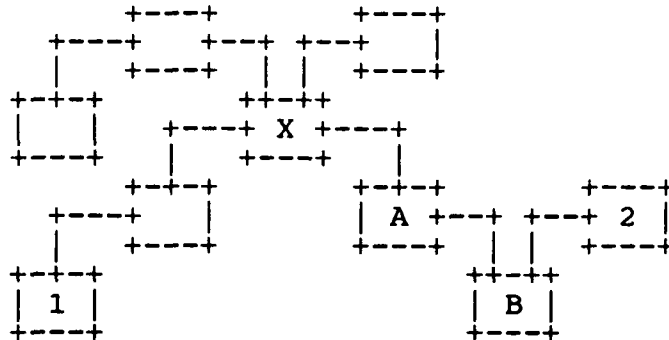


Figure 3-9. Entire Object Adjustments

If objects 1 and 2 become directly related, an adjustment will have to be made to align these two objects along the same diagonal. The Layout system will find the commonly related object with the most relationships and relocate the entire group of objects to the appropriate diagonal. Figure 3-10 shows the outcome of the adjustment.

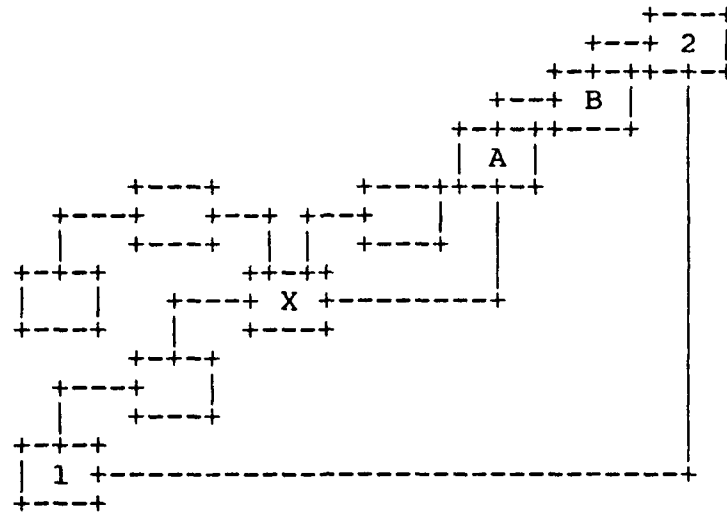


Figure 3-10. Object Diagonal

In this example, object 2 as well as the intermediate objects will be moved as shown in the diagram. Object X is the commonly related object with the most relationships.

After all objects have been placed in the chart space in this manner, the Layout system will begin a refinement process which will distribute the objects more evenly. Each object will be searched for free corners which will be used to migrate a group of independently related objects from a corner of that object where more than one such group is located. Figure 3-11 shows the upper right-hand corner of object X as a corner containing more than one independent group of related objects.

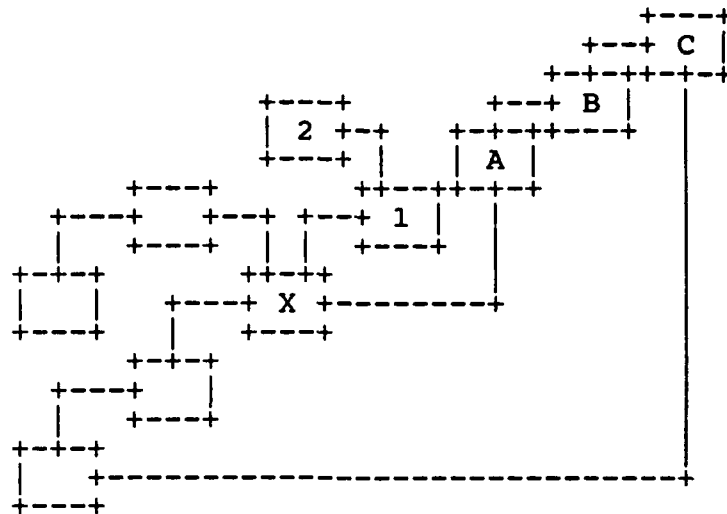


Figure 3-11. Independent group of Objects

Group [1,2] or group [A,B,C] can be moved to the lower right-hand corner of object X. In this example, objects [1,2] will be moved. Figure 3-12 shows the result of this realignment.

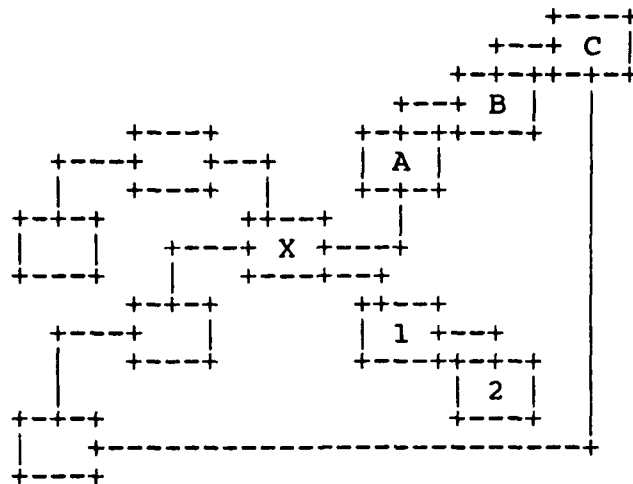


Figure 3-12. Object X

The next step in the refinement process is to migrate as many objects as possible from diagonals to the sides of objects. Objects and relationships governed by archetype definition characteristics will be adjusted first. After that, an additional examination is made to determine whether further refinements of this type are possible.

If moving a relationship to a side would cause another relationship to increase its number of line bends to two, the migration of the relationship will not occur. In Figure 3-13, object 4 can be moved to either the top or the right side of object 3. However, due to the increase in the number of line bends in relationships A and B, object 4 will not be moved to the top or right side of object 3. Moving object 4 to the left side of object 3 would be a valid move.

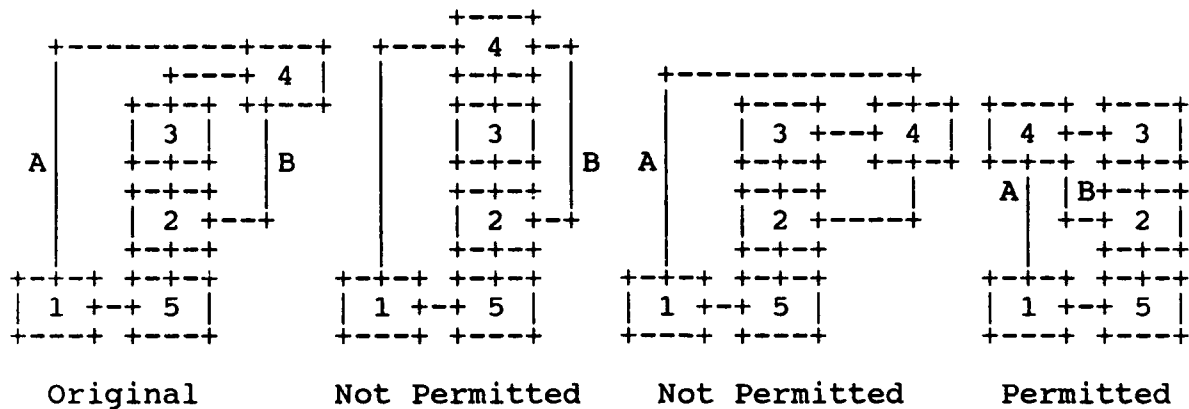


Figure 3-13. Permitted Adjustments

The final step in the layout process is determining the line lengths for the relationships of the chart. Line lengths are calculated from the inside out, that is, the lines which do not derive their lengths from the length of other line calculations are calculated first. This insures an efficient length calculation process in that the number of line length recalculations will be held to a minimum. The process begins by locating the object which best satisfies the chart orientation characteristics. For a chart with a horizontal orientation, the object with only relationships originating from its bottom side is selected. For a chart with vertical orientation, the object with only relationships originating from its right side is selected. If these objects did not exist in the chart, the object which came closest to satisfying the criteria is used as the starting object.

Length calculations are made by first traversing the chart to the extremity objects, marking the relationships traveled so that they may be identified as being calculated. (An extremity object is one that has no other relationships associated with it other than the one that was used to get to it.) When an extremity object is reached, the vertical and horizontal distances are updated. Relationships that were marked as "being calculated" are marked "calculated". The object width and depth and the minimum spacing between objects from the chart characteristic are taken into consideration when determining these lengths.

Figure 3-14 is provided to clarify the rules of determining line lengths.

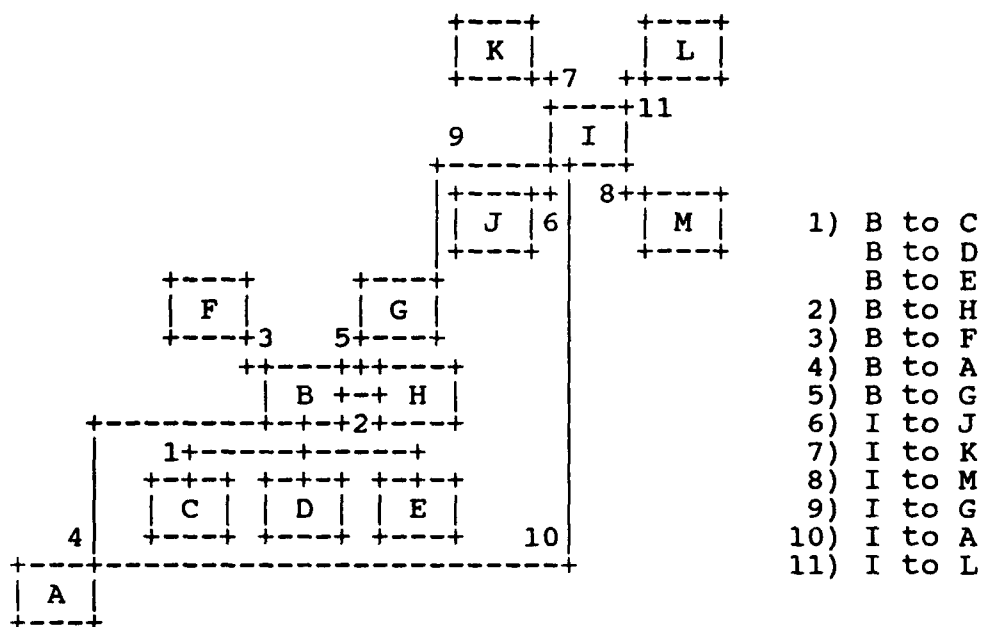


Figure 3-14. Line Length Rules

The relationships are listed in the order in which they are to be processed. In this example, the initial layout preference is "leftup".

### 3.2.5 Chart Display

PRNCHT displays the drawn chart. The chart will be displayed on the device specified by the device name parameter. The chart will be broken down into pages according to the pagination style specified in the pagination style parameter. The page size will be determined by the device page size.

The two pagination styles provided are physical and logical. The physical style sections the chart into pages without regard as to where the page breaks occur. These pages can later be bound together to reconstruct the entire chart. The logical pagination style will break the chart down into logical pages with off-page connector symbols to identify the continuation points. This style is useful when charts are to be stored in binders.

### 3.2.6 Chart Knowledge Base

The Layout Optimization System will make use of a knowledge base for charts. The chart knowledge base will contain information pertaining to:

- o The formation of the chart.
- o The object icon form names and restrictions pertaining to the placement of connector lines with respect to the object.
- o The relation originator, terminator, and complex icon form names, connector line style description and relation placement restrictions.

#### 3.2.6.1 Knowledge Base Utility Program

The knowledge base utility program is accessed through the IISS function screen. The following sections explain how the program is used.

##### 3.2.6.1.1 Chart Description

When the knowledge base utility program is selected from the IISS function screen, the chart definition form (Figure 3-15) will be displayed.



Layout Optimization System Chart Description		
Chart Name _____		
Chart Orientation	Initial Layout Preference	
<input type="checkbox"/> Vertical	<input type="checkbox"/> Left Downward	
<input type="checkbox"/> Horizontal	<input type="checkbox"/> Left Upward	
Minimum spacing between objects: _____ characters.		
<Enter> Enter	<PF5> Delete	<PF7> Objects
<Quit> Terminate		<PF8> Relations

Figure 3-15. Chart Description

The Chart Name field is a ten character field used to specify the chart archetype name. The Chart Orientation is selected by placing a non-blank character beside either Vertical or Horizontal. The Initial Layout Preference is selected in the same manner. A non-blank character is placed beside Left Downward or Left Upward. The Minimum Spacing Between Objects is an optional numeric field. Values from 1 through 15 are valid. The functions and function keys are displayed at the bottom of the form. <Enter> on the keypad is used to enter the data. <Quit> is used to terminate the utility. If <PF5> is selected the chart description will be deleted. The Object and Relation description forms can be accessed by pressing <PF6> or <PF7> respectively. The data should be entered into the fields as required and <Enter> pressed to input the values.

#### 3.2.6.1.2 Object Description

When the user selects the Objects function key the form illustrated in Figure 3-16 will be displayed.

Chart Name		Layout Optimization System Object Description	
Connector Placement Criteria			
Objects	Icon Names	Originating: No Criteria	
+	+	Bottom	Top Left Right
+	+	Bottom Right	Corner
+	+	Bottom Left	Corner
+	+	Top Right	Corner
+	+	Top Left	Corner
		Terminating: No Criteria	
+	+	Bottom	Top Left Right
+	+	Bottom Right	Corner
+	+	Bottom Left	Corner
+	+	Top Right	Corner
+	+	Top Left	Corner
<Enter> Enter		<PF5> Delete	
<Quit> Terminate		<PF6> Chart	
		<PF8> Relations	

Figure 3-16. Object Description

The chart name will be displayed in the upper left-hand corner of the form. The name will not be modifiable. Objects currently defined for this chart will be displayed in the scrollable area labeled Objects. The Icon names will be displayed adjacent to the objects. To define a new object, enter the object name into the first available blank slot, leaving the cursor in the slot, and press <Enter>. This will highlight the new object name and make that object the current object. The remainder of the object definition can be supplied at this time. The Originating and Terminating connector placement restrictions are selected by placing a non-blank character beside the appropriate item. When the description is complete, press <Enter> and the description will be assigned to the object highlighted. To delete an object description, place the cursor on the object name to be deleted and press <PF5>. To modify an object description, place the cursor on the object name to be modified and press <Enter>. The description for this object will be displayed. When the modifications are completed, press <Enter> to update the object description.

### 3.2.6.1.3 Relation Description

When the user selects the Relations function key the form illustrated in Figure 3-17 will be displayed.

Chart Name	Layout Optimization System Relation Description	
	Icon Names	Line Style
Relations	Origination	
+-----+	Termination	
+-----+	Complex	
+-----+	Connector Placement Criteria	
+-----+	Origination: _No Criteria	
+-----+	_Bottom _Top _Left _Right	
+-----+	Termination: _No Criteria	
+-----+	_Bottom _Top _Left _Right	
+-----+	Special Characteristics	
+-----+	Two Bend Line Slope: _Up _Down _Left _Right	
+-----+	Combine Relations: _Yes _No	
<Enter> Enter	<PF5> Delete	<PF6> Chart
<Quit> Terminate		<PF7> Objects

Figure 3-17. Relation Description

The chart name will be displayed in the upper left-hand corner of the form. The name will not be modifiable. Relations currently defined for this chart will be displayed in the scrollable area labeled "Relations". To define a new relation, enter the relation name into the first available blank slot, leaving the cursor in the slot, and press <Enter>. This will highlight the new relation name and make it the current relation. The remainder of the relation definition can be supplied at this time. The Icon Names for the origination, termination and complex symbols are the form names that were specified at form definition. The Line Style field is used to describe the connector line style. The line styles are defined by the keywords SOLID, DASHED, DOTTED AND DASH\_DOTTED. The Placement Restrictions are specified for Origination and Termination by placing a non-blank character by the appropriate choices. The special two bend line slope description, explained in Appendix A, is specified, if required, by placing a non-blank character beside the desired choice. When the description is complete, press <Enter> and the description will be assigned to the relation highlighted. To delete a relation description, place the cursor on the relation name to be deleted and press <PF5>. To modify an relation description, place the cursor on the relation name to be modified and press <Enter>. The description for this relation will be displayed. When the modifications are completed, press <Enter> to update the relation description.

#### 3.2.6.2 Knowledge Base Description

The Layout Optimization System Knowledge Base consists of three files contained in a directory pointed to by the logical assignment IISSKLIB. The file names are loscht.dat, losobj.dat and losrel.dat. These files will have an indexed-sequential file organization. The indexed-sequential file organization will be handled by COBOL input/output routines.

##### 3.2.6.2.1 Charts File Layout

The loscht.dat file contains the information related to a chart description. All chart descriptions will be contained in this file. The key for this file will be chart name. The record layout for this file is:

<u>Positions</u>	<u>Field</u>	<u>Data Type</u>
1 - 10	chart name	character
11 - 20	object orientation	character
21 - 30	initial layout structure	character
31 - 32	minimum spacing	numeric

#### 3.2.6.2.2 Objects File Layout

The losobj.dat file contains the information related to an object description. All object descriptions will be contained in this file. The key for this file will be object name and chart name. The record layout for this file is:

<u>Positions</u>	<u>Field</u>	<u>Data Type</u>
1 - 10	object name	character
11 - 20	chart name	character
21 - 30	originate connections	character
31 - 40	terminate connections	character
41 - 50	object icon form	character

#### 3.2.6.2.3 Relations File Layout

The losrel.dat file contains the information related to a relation description. All relation descriptions will be contained in this file. The key for this file will be relation name and chart name. The record layout for this file is:

<u>Positions</u>	<u>Field</u>	<u>Data Type</u>
1 - 10	relation name	character
11 - 20	chart name	character
21 - 30	origination icon form	character
31 - 40	termination icon form	character
41 - 50	complex icon form	character
51 - 60	connector line style	character
61 - 70	origination location	character
71 - 80	termination location	character
81 - 90	two bend line	character
91 - 100	combine connections	character

### 3.3 Performance Requirements

#### 3.3.1 Programming Methods

The Layout System will be provided as a set of callable services accessible from within an application program.

### 3.3.2 Program Organization

The Layout Optimization System will be developed in a modular way and will comprise functionally cohesive modules. It will be configured as an extension to the Application Interface.

### 3.3.3 Expandability

The Layout Optimization System will be developed with a view to admitting future possible extensions in terms of alternative interfaces other than a callable application interface. Also, to the extent feasible, constraints which are imposed in the current implementation will not be incorporated as fundamental and mandatory design limitations but as restrictions which may be relaxed if warranted in the future.

### 3.3.4 Error Recovery

The Layout Optimization System will recover from errors which may arise in its use and return appropriate error codes to indicate the nature and cause of the error. In addition messages will be displayed to the user in the message line.

### 3.4 Data Base Requirements

The Layout Optimization System has no database requirements other than the chart knowledge base, which has been specified in Section 3.2.6.

### 3.5 Adaptation Requirements

The Layout Optimization System will be required to operate in those environments hosting IISS. No modifications to applications shall be necessary in order to rehost to a different host processor provided the published coding guidelines are followed.

## SECTION 4

### QUALITY ASSURANCE PROVISIONS

#### 4.1 Introduction and Definition

"Testing" is a systematic process that may be preplanned and explicitly stated. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error.

"Antibugging" is defined as the philosophy of writing programs in such a way as to make bugs less likely to occur and when they do occur, to make them more noticeable to the programmer and the user. As much error checking as is practical and possible in each routine should be performed.

#### 4.2 Computer Programming Test and Evaluation

The quality assurance provisions for test consists of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests are performed by the design team. Structured design, design walk-through and the incorporation of "antibugging" facilitate this testing by exposing and addressing problem areas before they become coded "bugs".

Each function is tested separately, then the entire subsystem is tested as a unit. Development and testing are done on the Air Force VAX.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the Integrated Information Support System (IISS) Test Bed site located at Arizona State University, Tempe, Arizona. The software associated with each CPC released is delivered on a media which is compatible with the IISS Test Bed. The release is clearly identified and includes instructions on procedures to be followed for installation of the release. Integration with the other IISS CPCs is done on the IISS Test Bed on a scheduled basis.



## APPENDIX A

### Callable Routines Description

This appendix describes the routines that are the library of services provided which enable the application program to communicate with the facilities of Layout Optimization System.

INITFL - used to acquire and initialize the computer resources that will be required by the application program.

inputs: none

outputs: completion\_status, character, 5 bytes

TERMFL - used to relinquish the computer resources acquired during the use of the system.

inputs: none

outputs: completion\_status, character, 5 bytes

DCHART - describe a chart archetype.

inputs: chart\_archetype\_reference\_name,  
character, 10 bytes, used as the  
reference name for the chart  
archetype.

keyword\_list\_length, binary integer, 4  
bytes, specifies the length of the  
keyword\_list.

keyword\_list, character, a list of  
chart characteristic keywords,  
separated by commas. Valid keywords  
are:

vertical - provide vertical object  
orientation.  
horizontal - provide horizontal object  
orientation.  
leftdown - provide left downward  
initial layout structure.  
leftup - provide left upward  
initial layout structure.  
minspace=99- minimum spacing between  
objects.

outputs: completion\_status, character, 5 bytes

MAKCHT - make a chart instance from a chart archetype.

inputs: chart\_instance\_name, character, 10  
bytes

chart\_archetype\_reference\_name,  
character, 10 bytes

outputs: completion\_status, character, 5 bytes

DELCHT - deletes all object and relation definitions,  
and frees all acquired computer resources for  
the named chart instance.

inputs: chart\_instance\_name, character, 10  
bytes.

outputs: completion\_status, character, 5 bytes

DOBJTP - describe an object archetype.

inputs: chart\_archetype\_reference\_name,  
character, 10 bytes, the name of  
the chart archetype to which this  
description is to be assigned.

object\_archetype\_reference\_name,  
character, 10 bytes

object\_form\_name, character, 10 bytes,  
the name of the form to be used as the  
pictorial representation of this  
object.

keyword\_list\_length, binary integer, 4  
bytes, the length of the  
keyword\_list.

keyword\_list, character, a list of  
object characteristic keywords,  
separated by commas. Valid keywords  
are:

- psocbottom - all originating  
connections should be  
located at the bottom of  
the object.
- psoctop - all originating  
connections should be  
located at the top of the  
object.
- psocleft - all originating  
connections should be  
located to the left of  
the object.
- psocright - all originating  
connections should be  
located to the right of  
the object.
- cstcbottom - all terminating  
connections should be  
located at the bottom of  
the object.
- cstctop - all terminating  
connections should be  
located at the top of the  
object.
- cstcleft - all terminating  
connections should be  
located to the left of  
the object.
- cstcright - all terminating  
connections should be  
located to the right of  
the object.

pcocbright - all originating connections should be located on the bottom right-hand corner of the object.  
pcocbleft - all originating connections should be located on the bottom left-hand corner of the object.  
pcocuright - all originating connections should be located on the upper right-hand corner of the object.  
pcoculeft - all originating connections should be located on the upper left-hand corner of the object.  
cctcbright - all terminating connections should be located on the bottom right-hand corner of the object.  
cctcbleft - all terminating connections should be located on the bottom left-hand corner of the object.  
cctcuright - all terminating connections should be located on the upper right-hand corner of the object.  
cctculeft - all terminating connections should be located on the upper left-hand corner of the object.

outputs: completion\_status, character, 5 bytes

MAKOBJ - create an instance of the object archetype.

inputs: chart\_instance\_name, character, 10  
bytes, instance name of the chart to  
which this object is to be assigned.

object\_instance\_name, character, 10  
bytes, instance name for this object.

object\_archetype\_reference\_name,  
character, 10 bytes, reference name of  
the object archetype to use for this  
object.

outputs: completion\_status, character, 5 bytes

DELOBJ - deletes the named object instance from a  
chart instance.

inputs: chart\_instance\_name, character, 10  
bytes, instance name of the chart  
containing the object to be deleted.

object\_instance\_name, character, 10  
bytes, instance name of the object to  
be deleted.

outputs: completion\_status, character, 5 bytes

BEGCHT - changes the application program's display list  
to that of the chart identified. Required to  
enable the application program to have access  
to the forms used to pictorially represent the  
objects and relationships of the chart.

inputs: chart\_instance\_name, character, 10  
bytes

outputs: completion\_status, 5 bytes

ENDCHT - returns the application program's display list  
from the chart specified in the BEGCHT routine  
to the application program's original display  
list.

inputs: none

outputs: completion\_status, character, 5 bytes

DRELTP - describe a relation archetype.

inputs: chart\_archetype\_reference\_name,  
character, 10 bytes, reference name of  
the chart archetype to which this  
relation description is to be  
assigned.

relation\_archetype\_reference\_name,  
character, 10 bytes, reference name  
for this relation archetype  
description.

termination\_symbol\_form\_name,  
character, 10 bytes, the name of the  
form to be used as the pictorial  
representation of the termination  
symbol of this relation.

origination\_symbol\_form\_name,  
character, 10 bytes, the name of the  
form to be used as the pictorial  
representation of the origination  
symbol of this relation.

complex\_symbol\_form\_name, character,  
10 bytes, the name of the form to be  
used as the pictorial representation  
of a complex relation. (ie. an IDEF1X  
generic relation).

line\_style, character, 10 bytes, the  
stretchy line style definition for the  
connector line expressed as a keyword:

solid           - the line is to be  
                  represented by a solid  
                  line.

dashed          - the line is to be  
                  represented by a dashed  
                  line.

dotted          - the line is to be  
                  represented by a dotted  
                  line.

dash\_dot       - the line is to be  
                  represented by           alternating  
                  dashes and           dots.

keyword\_list\_length, binary integer, 4  
bytes, the length of the keyword\_list.

keyword\_list, character, a list of

DS 620344800  
30 September 1990

relation characteristic keywords,  
separated by commas. Valid keywords  
are:

osright	- relationship originates from the right-hand side of an object.
osleft	- relationship originates from the left-hand side of an object.
osbottom	- relationship originates from the bottom of an object.
ostop	- relationship originates from the top of an object.
tsright	- relationship terminates on the right-hand side of an object.
tsleft	- relationship terminates on the left-hand side of an object.
tsbottom	- relationship terminates on the bottom of an object.
tstop	- relationship terminates on the top of an object.
slopeup	- relationship has two bends and slopes upward (see Figure A-1).
slopedown	- relationship has two bends and slopes downward (see Figure A-1).
slopeleft	- relationship has two bends and slopes to the left (see Figure A-1).
sloperight	- relationship has two bends and slopes to the right (see Figure A-1).
combine	- pictorially combine relationships of this type, whenever possible, when the chart is drawn.

outputs: completion\_status, character, 5 bytes

MAKREL - create an instance of a relation archetype.



inputs: chart\_instance\_name, character, 10 bytes, instance name of the chart to which this relationship is to be assigned.

relationship\_instance\_name, character, 10 bytes, instance name for this relationship. This name must have a maximum length of nine characters. There must be at least one blank character at the end of the name. This is required so the origination, termination and complex form instances can be referenced by the application program using the relationship\_instance name appended with the characters "O", "T", or "C" for "O"rigination, "T"ermination or "C"omplex.

parent\_object\_instance\_name, character, 10 bytes, instance name of the parent or origination object.

child\_object\_instance\_name, character, 10 bytes, instance name of the child or termination object.

relation\_archetype\_reference\_name, character, 10 bytes, reference name of the relation archetype to use for the relationship.

label\_length, binary integer, 4 bytes, the length of the text string in the label parameter.

label, character, variable length, non-modifiable text string to be associated with the relationship.

outputs: completion\_status, character, 5 bytes

DELREL - delete a relationship instance from a chart.

inputs: chart\_instance\_name, character, 10  
bytes  
  
relationship\_instance\_name, character,  
10 bytes  
  
parent\_object\_instance\_name,  
character, 10 bytes, instance name of  
the parent or origination object of  
the relationship.  
  
child\_object\_instance\_name, character,  
10 bytes, instance name of the child  
or termination object of the  
relationship.  
  
outputs: completion\_status, character, 5 bytes

DRWCHT - layout the objects and relationships for a  
chart.

inputs: chart\_instance\_name, character, 10  
bytes  
  
outputs: completion\_status, character, 5 bytes

PRNCHT - display a chart.

inputs: chart\_instance\_name, character, 10  
bytes  
  
display\_device\_type, character, 10  
bytes, type of device to be used for  
the display (eg. LN03).  
  
display\_device\_name, character, 10  
bytes, name of the device to be used  
for the display.  
  
pagination\_style, character, 10 bytes,  
valid values are:

logical - provide logical  
          pagination.  
physical - provide physical  
          pagination.

outputs: completion\_status, character, 5 bytes

GINSNM - generate an instance name. This routine is provided to give the application program a means to create a unique instance name which can be used in the MAKCHT, MAKOBJ OR MAKREL routine calls. Use of this routine is not required.

inputs: none

outputs: instance\_name, character, 10 bytes,  
          the generated instance name. This  
          name will be nine characters and  
          unique.

completion\_status, character, 5 bytes

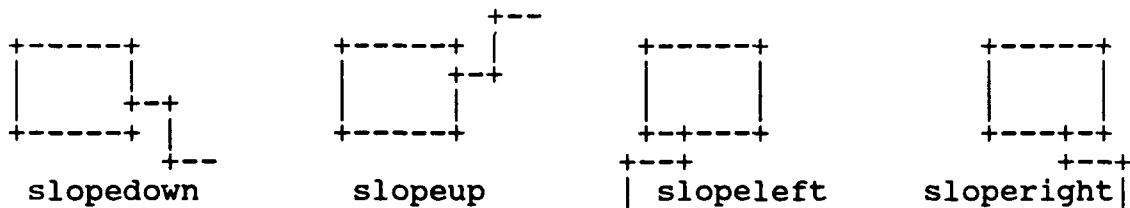


Figure A-1

## APPENDIX B

### Stretchy Lines

A "stretchy line" facility has been added to the Form Processor as an adjunct to the dynamic fields capability. This facility enables a line to stretch or shrink as dictated by the size of the field on which its definition depends. This appendix will preview the Form Processor syntax for stretchy line definition. Appendix C provides an example of how to use the stretchy line specification in a form definition.

#### Syntax

```
LINE line_name
  START [AT] location
  END [AT] location
  DISPLAY [AS] attribute_name
  [ STYLE style_string ]
  [ PROMPT location prompt_string ...]
```

The START AT and END AT specifications are the most important part of the line definition. It is from these specifications that the line gains its ability to stretch or shrink. If the start and end specifications are defined as fixed points, such as 1,1 to 1,5, then the line will not stretch because its start and end points are fixed and do not change. However, if the start and end specifications are defined as being relative to points that may move, such as the end points of a field definition, then the line will stretch or shrink as the field stretches, shrinks or increases in array size.

The following example shows how a line would stretch if the field length for the field the line depends upon were to change from ten to fifteen characters by the dynamic field call SETSIZ:

```
LINE AtoB
  START AT 1 below left of AB
  END   AT 1 below right of AB
```

```
ITEM AtoB
  at 1,1
  size 10
```

```
+-----+
+-----+
+-----+
```

before

```
+-----+
+-----+
+-----+
```

after

Style is used to define the line style. The line can be composed of characters from the ASCII character set or the line drawing character set. The style specification has three parts:

- [begin\_character] - specifies the character that will be used to identify the beginning of the line.
- [repeat\_specification] - specifies the character or characters that will be used as the line character.
- [end\_character] - specifies the character that will be used to show the end of the line.

If one character is specified for Style it is interpreted as the begin\_character. If two or more characters are specified for Style the last character is interpreted as the end character and all characters in between are interpreted as the repeating line specification.

Any character from the ASCII character set may be used to represent the begin, repeat and end characters. However, the characters used to describe components of the line drawing character set are a little more complex. To describe a horizontal solid line, the character sequence "\hs" would be used as the repeat character. The Style definition of "+\hs+" would create a line that appears as follows:

```
+-----+
```

The complete set of line drawing characters available for use in the stretchy line facility and their definition codes are:

<u>character</u>	<u>code</u>	<u>character</u>	<u>code</u>
-----	\hs		\vs
-+ 	\vd	-----+	\hd
+-- 	\vu	+-----	\hu
-+ 	\vr	-----+	\hr
 +-	\vl	+-----	\hl

If the Style clause is omitted, the default style will be solid vertical and horizontal lines drawn from the line drawing character set. If the display device does not support the line drawing character set, a horizontal line will be drawn with the "-" character and a vertical line will be drawn with the "|" character.

The following examples show some possible style specifications and the way they would be displayed. The examples assume a resultant line length of five characters.

<u>style specification</u>	<u>appearance</u>	
	<u>horizontal</u>	<u>vertical</u>
+++	+----+	+ - - - +
++	+----+	+   +
+	+-----	+       
o.x.o	o.x.o	o . x .

no style specified and  
display device does  
not support line drawing  
characters

-----

o

|

+\hs+

+----+

+  
-  
-  
+

\hl\hd

+----+

+--  
|  
---+

no style specified  
and device supports  
line drawing characters

-----

|

APPENDIX C  
Stretchy Line Example

The following example illustrates the use of stretchy line definitions to create a pictorial representation of an object. The object to be created is an IDEF1X independent entity.

create form entity

```
/*
 * entity name and number field
 */
    item name_field
        at 1,4
        size 20
        display as text
        domain (left)

/*
 * primary key field
 */
    item key_field      * v 0
        at 2 below name_field
        size 20
        display as text
        domain (left)

/*
 * attribute name field
 */
    item attr_field      * v 0
        at 2 below key_field
        display as text
        domain (left)
        size 20

/*
 * horizontal top border line
 */
    line hor_top_line
        start at -1 -2 relative to top left of key_field
        end   at -1  2 relative to top right of key_field
        display as text
        style +--+
```



```
/*
 * vertical left border line
 */
    line vert_left_line
        start at 1 below left of hor_top_line
        end   at 1 above left of hor_bot_line
        display as text
        style |

/*
 * vertical right border line
 */
    line vert_right_line
        start at 1 below right of hor_top_line
        end   at 1 above right of hor_bot_line
        display as text
        style |

/*
 * horizontal bottom border line
 */
    line hor_bot_line
        start at 1 -2 relative to bottom left of attr_field
        end   at 1  2 relative to bottom right of attr_field
        display as text
        style +--+

/*
 * horizontal line dividing key area from attribute area
 */
    line hor_key_attr_line
        start at 1 -1 relative to bottom left of key_field
        end   at 1  1 relative to bottom right of key_field
        display as text
        style --
```

If the object were displayed as defined, it would appear as follows:

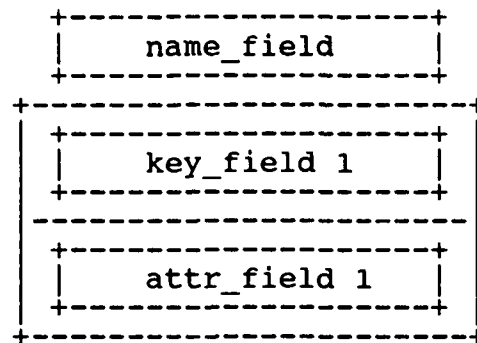


Figure C-1 Object As Defined

If, through Form Processor dynamic field calls, the `name_field` were shortened to twelve characters, the `key_field` length increased to twenty five characters and another element added to its array size, and the `attr_field` length increased to twenty five and another element added to its array size, the object, when displayed, would appear as follows:

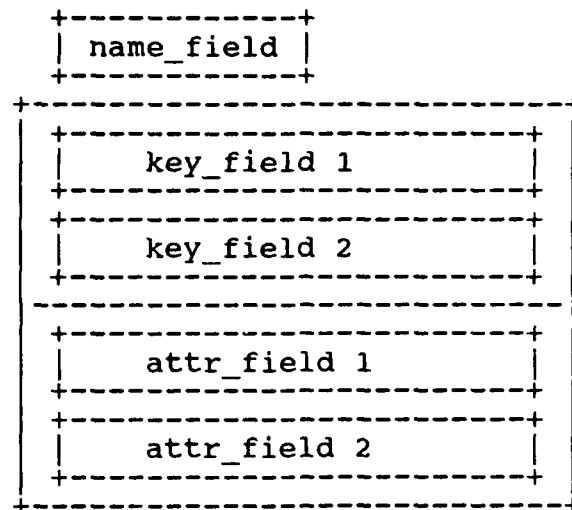


Figure C-2 Object After Change

The following examples illustrate how connector terminator symbols might be defined using FDL. These examples are for a few of the IDEF1X cardinality symbols:

```

/*
 * cardinality greater than
 * or equal to zero
 */

```

create form cardge0

```

prompt at 1,3 "***"
prompt at 2,2 "*****"
prompt at 3,3 "***"

```

if displayed

```

**
****
**

```

```

/*
 * cardinality greater than zero
 */

```

create form cardp

```
prompt at 1,3 "***"           **
prompt at 2,2 "*****"        **** P
prompt at 3,3 "***"           **
prompt at 2,7 "p"
```

```
/*
 * cardinality exactly n, where n
 * would be supplied by the application
 */
```

create form cardn

```
prompt at 1,3 "***"           **
prompt at 2,2 "*****"        **** 5
prompt at 3,3 "***"           **
```

```
item card_num
  at 2,7
  size 3
  display as text
  domain (left)
```

```
/*
 * categorization relationship
 * connector termination symbol
 */
```

create form generic

```
item discriminator
  at 1,20
  size 20
  display as text
  domain (left)
  prompt at left "("
  prompt at right ")"
```

(discriminator)  
-----  
-----

```
line hline1
  start at 1 -5 relative to
    bottom left of discriminator
  end   at 1 5 relative to
    bottom right of discriminator
  display as text
```

line hline2  
start at 2 -5 relative to  
bottom left of discriminator  
end at 2 5 relative to  
bottom right of discriminator